```
#Importing the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

data = pd.read_csv('uber.csv')

data
```

	Unnamed: 0	key	fare_amount	\
0	24238194	2015-05-07 19:52:06.0000003	7.5	
1	27835199	2009-07-17 20:04:56.0000002	7.7	
2	44984355	2009-08-24 21:45:00.00000061	12.9	
3	25894730	2009-06-26 08:22:21.0000001	5.3	
4	17610152	2014-08-28 17:47:00.000000188	16.0	
...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	

	pickup_datetime	pickup_longitude	pickup_latitude	\
0	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
...	...	...	...	
199995	2012-10-28 10:49:00 UTC	-73.987042	40.739367	
199996	2014-03-14 01:09:00 UTC	-73.984722	40.736837	
199997	2009-06-29 00:42:00 UTC	-73.986017	40.756487	
199998	2015-05-20 14:56:25 UTC	-73.997124	40.725452	
199999	2010-05-15 04:08:00 UTC	-73.984395	40.720077	

	dropoff_longitude	dropoff_latitude	passenger_count
0	-73.999512	40.723217	1
1	-73.994710	40.750325	1
2	-73.962565	40.772647	1
3	-73.965316	40.803349	3
4	-73.973082	40.761247	5
...	...	...	...
199995	-73.986525	40.740297	1
199996	-74.006672	40.739620	1

```
199997         -73.858957         40.692588                2
199998         -73.983215         40.695415                1
199999         -73.985508         40.768793                1

[200000 rows x 9 columns]
```

```
data.head()
```

```
   Unnamed: 0                            key  fare_amount  \
0    24238194    2015-05-07 19:52:06.0000003          7.5
1    27835199    2009-07-17 20:04:56.0000002          7.7
2    44984355   2009-08-24 21:45:00.00000061         12.9
3    25894730    2009-06-26 08:22:21.0000001          5.3
4    17610152  2014-08-28 17:47:00.000000188         16.0

           pickup_datetime  pickup_longitude  pickup_latitude  \
0  2015-05-07 19:52:06 UTC        -73.999817        40.738354
1  2009-07-17 20:04:56 UTC        -73.994355        40.728225
2  2009-08-24 21:45:00 UTC        -74.005043        40.740770
3  2009-06-26 08:22:21 UTC        -73.976124        40.790844
4  2014-08-28 17:47:00 UTC        -73.925023        40.744085

   dropoff_longitude  dropoff_latitude  passenger_count
0         -73.999512         40.723217                1
1         -73.994710         40.750325                1
2         -73.962565         40.772647                1
3         -73.965316         40.803349                3
4         -73.973082         40.761247                5
```

```
data.tail()
```

```
        Unnamed: 0                            key  fare_amount  \
199995    42598914  2012-10-28 10:49:00.00000053          3.0
199996    16382965   2014-03-14 01:09:00.0000008          7.5
199997    27804658  2009-06-29 00:42:00.00000078         30.9
199998    20259894   2015-05-20 14:56:25.0000004         14.5
199999    11951496  2010-05-15 04:08:00.00000076         14.1

                pickup_datetime  pickup_longitude  pickup_latitude  \
199995  2012-10-28 10:49:00 UTC        -73.987042        40.739367
199996  2014-03-14 01:09:00 UTC        -73.984722        40.736837
199997  2009-06-29 00:42:00 UTC        -73.986017        40.756487
199998  2015-05-20 14:56:25 UTC        -73.997124        40.725452
199999  2010-05-15 04:08:00 UTC        -73.984395        40.720077

        dropoff_longitude  dropoff_latitude  passenger_count
199995         -73.986525         40.740297                1
199996         -74.006672         40.739620                1
199997         -73.858957         40.692588                2
199998         -73.983215         40.695415                1
199999         -73.985508         40.768793                1
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

```
data.describe()
```

```
         Unnamed: 0    fare_amount  pickup_longitude  pickup_latitude
\
count  2.000000e+05  200000.000000     200000.000000    200000.000000

mean   2.771250e+07      11.359955        -72.527638        39.935885

std    1.601382e+07       9.901776         11.437787         7.720539

min    1.000000e+00     -52.000000      -1340.648410       -74.015515

25%    1.382535e+07       6.000000        -73.992065        40.734796

50%    2.774550e+07       8.500000        -73.981823        40.752592

75%    4.155530e+07      12.500000        -73.967154        40.767158

max    5.542357e+07     499.000000         57.418457      1644.421482


       dropoff_longitude  dropoff_latitude  passenger_count
count      199999.000000     199999.000000    200000.000000
mean          -72.525292         39.923890         1.684535
std            13.117408          6.794829         1.385997
min         -3356.666300       -881.985513         0.000000
25%           -73.991407         40.733823         1.000000
50%           -73.980093         40.753042         1.000000
75%           -73.963658         40.768001         2.000000
max          1153.572603        872.697628       208.000000
```

```
data.shape
```

```
(200000, 9)
```

```
data.columns
```

```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

```
data = data.drop(['Unnamed: 0', 'key'], axis = 1)
```

```
data.shape
```

```
(200000, 7)
```

```
data
```

```
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count
0             40.738354         -73.999512         40.723217
1
1             40.728225         -73.994710         40.750325
1
2             40.740770         -73.962565         40.772647
1
3             40.790844         -73.965316         40.803349
3
4             40.744085         -73.973082         40.761247
5
...                 ...                ...                ...
...
199995        40.739367         -73.986525         40.740297
1
199996        40.736837         -74.006672         40.739620
1
199997        40.756487         -73.858957         40.692588
2
199998        40.725452         -73.983215         40.695415
```

```
1
199999        40.720077         -73.985508         40.768793
1

[200000 rows x 7 columns]
```

```
data['month'] = data['pickup_datetime']
data
```

```
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count  \
0             40.738354         -73.999512         40.723217
1
1             40.728225         -73.994710         40.750325
1
2             40.740770         -73.962565         40.772647
1
3             40.790844         -73.965316         40.803349
3
4             40.744085         -73.973082         40.761247
5
...                 ...                ...                ...
...
199995        40.739367         -73.986525         40.740297
1
199996        40.736837         -74.006672         40.739620
1
199997        40.756487         -73.858957         40.692588
2
199998        40.725452         -73.983215         40.695415
1
199999        40.720077         -73.985508         40.768793
1

                          month
0       2015-05-07 19:52:06 UTC
1       2009-07-17 20:04:56 UTC
```

```
2       2009-08-24 21:45:00 UTC
3       2009-06-26 08:22:21 UTC
4       2014-08-28 17:47:00 UTC
...                         ...
199995  2012-10-28 10:49:00 UTC
199996  2014-03-14 01:09:00 UTC
199997  2009-06-29 00:42:00 UTC
199998  2015-05-20 14:56:25 UTC
199999  2010-05-15 04:08:00 UTC

[200000 rows x 8 columns]
```

```
data['month'] = data['month'].str.slice(start = 5, stop = 7)
data
```

```
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count  \
0             40.738354         -73.999512         40.723217
1
1             40.728225         -73.994710         40.750325
1
2             40.740770         -73.962565         40.772647
1
3             40.790844         -73.965316         40.803349
3
4             40.744085         -73.973082         40.761247
5
...                 ...                ...               ...
...
199995        40.739367         -73.986525         40.740297
1
199996        40.736837         -74.006672         40.739620
1
199997        40.756487         -73.858957         40.692588
2
199998        40.725452         -73.983215         40.695415
1
```

```
199999        40.720077         -73.985508         40.768793
1

       month
0         05
1         07
2         08
3         06
4         08
...      ...
199995    10
199996    03
199997    06
199998    05
199999    05

[200000 rows x 8 columns]
```

```
data['hour'] = data['pickup_datetime']
data
```

```
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                      ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count  \
0             40.738354         -73.999512         40.723217
1
1             40.728225         -73.994710         40.750325
1
2             40.740770         -73.962565         40.772647
1
3             40.790844         -73.965316         40.803349
3
4             40.744085         -73.973082         40.761247
5
...                 ...                ...               ...
...
199995        40.739367         -73.986525         40.740297
1
```

```
199996         40.736837          -74.006672          40.739620
1
199997         40.756487          -73.858957          40.692588
2
199998         40.725452          -73.983215          40.695415
1
199999         40.720077          -73.985508          40.768793
1
```

```
       month                        hour
0         05  2015-05-07 19:52:06 UTC
1         07  2009-07-17 20:04:56 UTC
2         08  2009-08-24 21:45:00 UTC
3         06  2009-06-26 08:22:21 UTC
4         08  2014-08-28 17:47:00 UTC
...      ...                        ...
199995    10  2012-10-28 10:49:00 UTC
199996    03  2014-03-14 01:09:00 UTC
199997    06  2009-06-29 00:42:00 UTC
199998    05  2015-05-20 14:56:25 UTC
199999    05  2010-05-15 04:08:00 UTC

[200000 rows x 9 columns]
```

```
data['hour'] = data['hour'].str.slice(start = 11, stop = 13)
data
```

```
        fare_amount          pickup_datetime  pickup_longitude  \
0               7.5  2015-05-07 19:52:06 UTC        -73.999817
1               7.7  2009-07-17 20:04:56 UTC        -73.994355
2              12.9  2009-08-24 21:45:00 UTC        -74.005043
3               5.3  2009-06-26 08:22:21 UTC        -73.976124
4              16.0  2014-08-28 17:47:00 UTC        -73.925023
...             ...                        ...               ...
199995          3.0  2012-10-28 10:49:00 UTC        -73.987042
199996          7.5  2014-03-14 01:09:00 UTC        -73.984722
199997         30.9  2009-06-29 00:42:00 UTC        -73.986017
199998         14.5  2015-05-20 14:56:25 UTC        -73.997124
199999         14.1  2010-05-15 04:08:00 UTC        -73.984395

        pickup_latitude  dropoff_longitude  dropoff_latitude
passenger_count  \
0             40.738354         -73.999512         40.723217
1
1             40.728225         -73.994710         40.750325
1
2             40.740770         -73.962565         40.772647
1
3             40.790844         -73.965316         40.803349
3
```

```
4              40.744085         -73.973082         40.761247
5
...                  ...                ...                ...
...
199995         40.739367         -73.986525         40.740297
1
199996         40.736837         -74.006672         40.739620
1
199997         40.756487         -73.858957         40.692588
2
199998         40.725452         -73.983215         40.695415
1
199999         40.720077         -73.985508         40.768793
1

       month hour
0         05   19
1         07   20
2         08   21
3         06   08
4         08   17
...      ...  ...
199995    10   10
199996    03   01
199997    06   00
199998    05   14
199999    05   04

[200000 rows x 9 columns]
```

```
data = data.drop(['pickup_datetime'], axis = 1)
data
```

```
        fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
0               7.5        -73.999817        40.738354         -
73.999512
1               7.7        -73.994355        40.728225         -
73.994710
2              12.9        -74.005043        40.740770         -
73.962565
3               5.3        -73.976124        40.790844         -
73.965316
4              16.0        -73.925023        40.744085         -
73.973082
...             ...               ...               ...
...
199995          3.0        -73.987042        40.739367         -
73.986525
199996          7.5        -73.984722        40.736837         -
```

```
74.006672
199997         30.9        -73.986017        40.756487         -
73.858957
199998         14.5        -73.997124        40.725452         -
73.983215
199999         14.1        -73.984395        40.720077         -
73.985508

        dropoff_latitude  passenger_count month hour
0              40.723217                1    05   19
1              40.750325                1    07   20
2              40.772647                1    08   21
3              40.803349                3    06   08
4              40.761247                5    08   17
...                  ...              ...   ...  ...
199995         40.740297                1    10   10
199996         40.739620                1    03   01
199997         40.692588                2    06   00
199998         40.695415                1    05   14
199999         40.768793                1    05   04

[200000 rows x 8 columns]

data.describe()

         fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
count  200000.000000     200000.000000    200000.000000
199999.000000
mean       11.359955        -72.527638        39.935885         -
72.525292
std         9.901776         11.437787         7.720539
13.117408
min       -52.000000      -1340.648410       -74.015515       -
3356.666300
25%         6.000000        -73.992065        40.734796         -
73.991407
50%         8.500000        -73.981823        40.752592         -
73.980093
75%        12.500000        -73.967154        40.767158         -
73.963658
max       499.000000         57.418457      1644.421482
1153.572603

       dropoff_latitude  passenger_count
count     199999.000000    200000.000000
mean          39.923890         1.684535
std            6.794829         1.385997
min         -881.985513         0.000000
25%           40.733823         1.000000
```

```
50%           40.753042         1.000000
75%           40.768001         2.000000
max          872.697628       208.000000
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   fare_amount        200000 non-null  float64
 1   pickup_longitude   200000 non-null  float64
 2   pickup_latitude    200000 non-null  float64
 3   dropoff_longitude  199999 non-null  float64
 4   dropoff_latitude   199999 non-null  float64
 5   passenger_count    200000 non-null  int64  
 6   month              200000 non-null  object 
 7   hour               200000 non-null  object 
dtypes: float64(5), int64(1), object(2)
memory usage: 12.2+ MB
```

```
data["dropoff_longitude"]=
data["dropoff_longitude"].fillna(data['dropoff_longitude'].mean())
data["dropoff_latitude"]=
data["dropoff_latitude"].fillna(data['dropoff_latitude'].mean())

# data.dropna(inplace=True)------- Drop complete row

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   fare_amount        200000 non-null  float64
 1   pickup_longitude   200000 non-null  float64
 2   pickup_latitude    200000 non-null  float64
 3   dropoff_longitude  200000 non-null  float64
 4   dropoff_latitude   200000 non-null  float64
 5   passenger_count    200000 non-null  int64  
 6   month              200000 non-null  object 
 7   hour               200000 non-null  object 
dtypes: float64(5), int64(1), object(2)
memory usage: 12.2+ MB
```

```
def haversine (lon_1, lon_2, lat_1, lat_2):
    lon_1, lon_2, lat_1, lat_2 = map(np.radians, [lon_1, lon_2, lat_1,
lat_2])
    diff_lon = lon_2 - lon_1
```

```
    diff_lat = lat_2 - lat_1
    km = 2 * 6371 * np.arcsin(np.sqrt(np.sin(diff_lat/2.0)**2 +
np.cos(lat_1) * np.cos(lat_2) * np.sin(diff_lon/2.0)**2))
    return km

data['distance'] =
haversine(data['pickup_longitude'],data['dropoff_longitude'],data['pic
kup_latitude'],data['dropoff_latitude'])

data.describe()
```

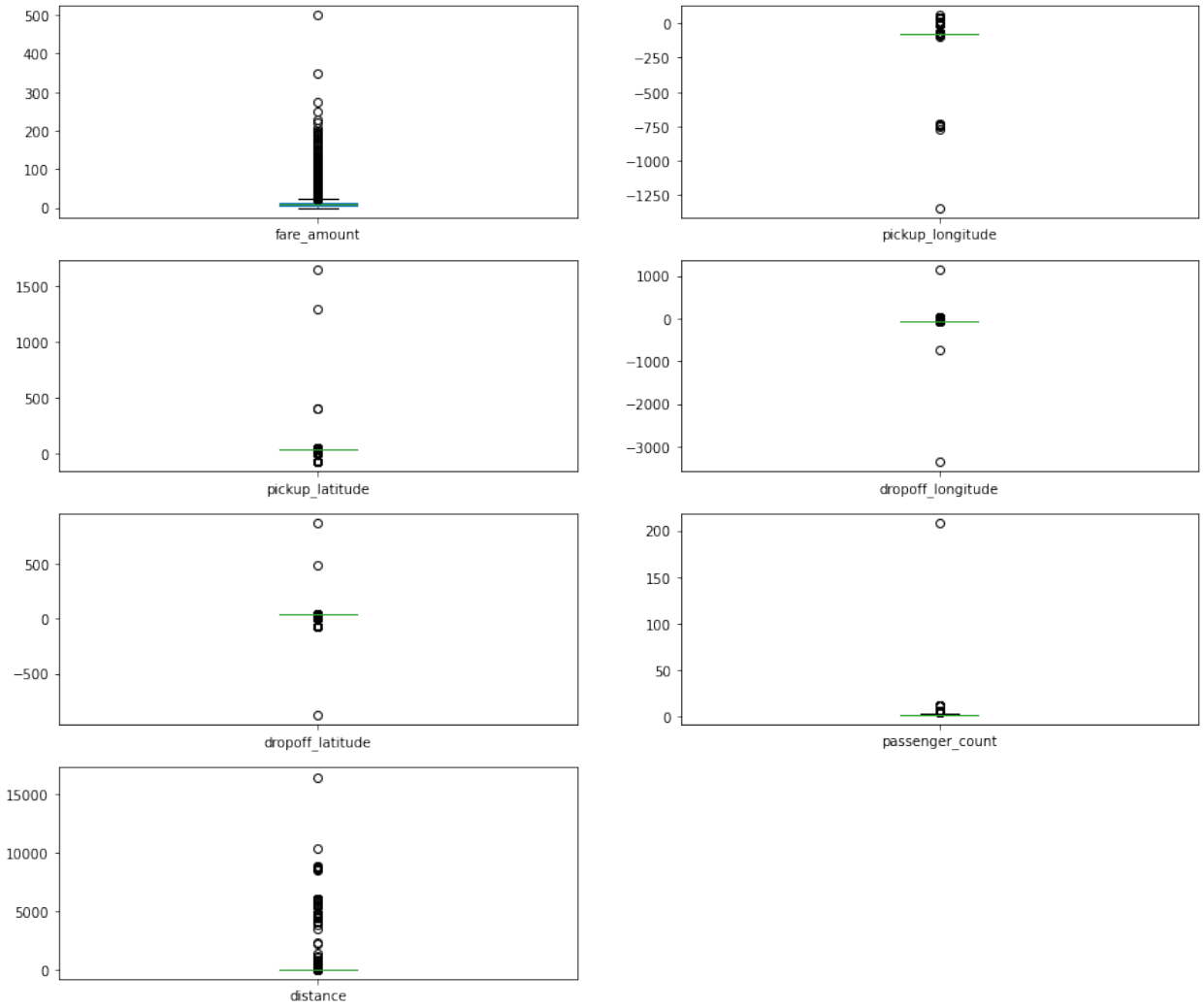
```
         fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
count  200000.000000     200000.000000    200000.000000
200000.000000
mean       11.359955        -72.527638        39.935885          -
72.525292
std         9.901776         11.437787         7.720539
13.117375
min       -52.000000      -1340.648410       -74.015515        -
3356.666300
25%         6.000000        -73.992065        40.734796          -
73.991407
50%         8.500000        -73.981823        40.752592          -
73.980093
75%        12.500000        -73.967154        40.767158          -
73.963658
max       499.000000         57.418457      1644.421482
1153.572603

       dropoff_latitude  passenger_count       distance
count     200000.000000    200000.000000  200000.000000
mean          39.923890         1.684535      20.856014
std            6.794812         1.385997     382.963800
min         -881.985513         0.000000       0.000000
25%           40.733823         1.000000       1.215222
50%           40.753042         1.000000       2.121005
75%           40.768001         2.000000       3.875248
max          872.697628       208.000000   16409.239135
```

```
data.replace(to_replace = 0, value = data['passenger_count'].mean(),
inplace=True)
data.replace(to_replace = 0, value = data['distance'].mean(),
inplace=True)
data[data['fare_amount'] <= 0] = data['fare_amount'].mean()

data.describe()
```

```
         fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
count  200000.000000     200000.000000    200000.000000
```

```
200000.000000
mean       11.362407        -72.488497        39.965242         -
72.486713
std         9.896653         11.666136         7.562049
13.314895
min         0.010000      -1340.648410       -74.015515       -
3356.666300
25%         6.000000        -73.992065        40.734785         -
73.991407
50%         8.500000        -73.981821        40.752590         -
73.980091
75%        12.500000        -73.967148        40.767157         -
73.963653
max       499.000000         57.418457      1644.421482
1153.572603

       dropoff_latitude  passenger_count       distance
count     200000.000000    200000.000000  200000.000000
mean          39.953240         1.691287      20.860885
std            6.614871         1.385145     382.473053
min         -881.985513         1.000000       0.000084
25%           40.733817         1.000000       1.305244
50%           40.753040         1.000000       2.121431
75%           40.767999         2.000000       3.876122
max          872.697628       208.000000   16409.239135

data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   fare_amount        200000 non-null  float64
 1   pickup_longitude   200000 non-null  float64
 2   pickup_latitude    200000 non-null  float64
 3   dropoff_longitude  200000 non-null  float64
 4   dropoff_latitude   200000 non-null  float64
 5   passenger_count    200000 non-null  float64
 6   month              200000 non-null  object
 7   hour               200000 non-null  object
 8   distance           200000 non-null  float64
dtypes: float64(7), object(2)
memory usage: 13.7+ MB

data

        fare_amount  pickup_longitude  pickup_latitude
dropoff_longitude  \
0               7.5        -73.999817        40.738354         -
```

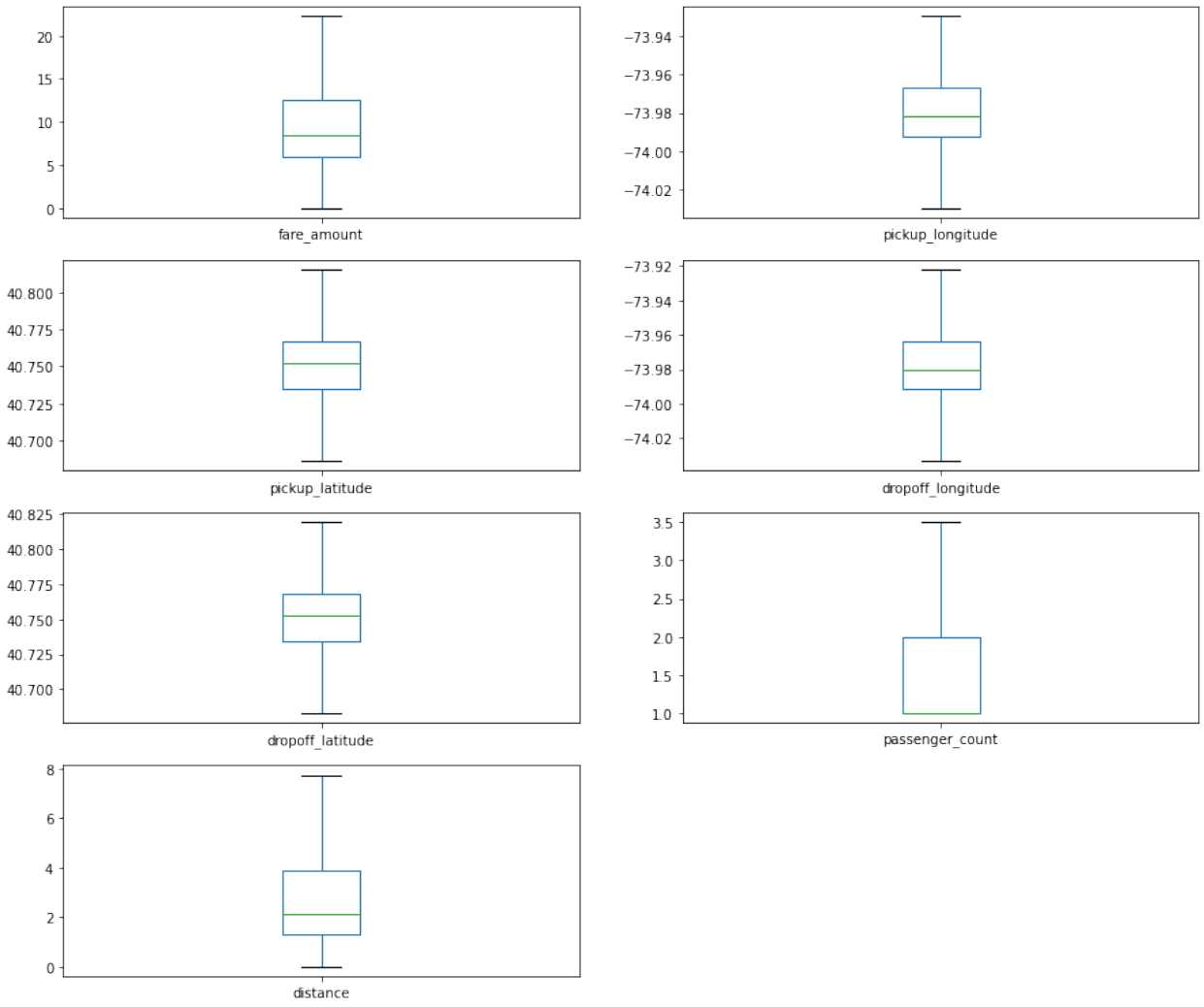
```
73.999512
1                7.7        -73.994355        40.728225         -
73.994710
2               12.9        -74.005043        40.740770         -
73.962565
3                5.3        -73.976124        40.790844         -
73.965316
4               16.0        -73.925023        40.744085         -
73.973082
...              ...               ...               ...
...
199995           3.0        -73.987042        40.739367         -
73.986525
199996           7.5        -73.984722        40.736837         -
74.006672
199997          30.9        -73.986017        40.756487         -
73.858957
199998          14.5        -73.997124        40.725452         -
73.983215
199999          14.1        -73.984395        40.720077         -
73.985508

        dropoff_latitude  passenger_count month hour   distance
0              40.723217              1.0    05   19   1.683323
1              40.750325              1.0    07   20   2.457590
2              40.772647              1.0    08   21   5.036377
3              40.803349              3.0    06   08   1.661683
4              40.761247              5.0    08   17   4.475450
...                  ...              ...   ...  ...        ...
199995         40.740297              1.0    10   10   0.112210
199996         40.739620              1.0    03   01   1.875050
199997         40.692588              2.0    06   00  12.850319
199998         40.695415              1.0    05   14   3.539715
199999         40.768793              1.0    05   04   5.417783

[200000 rows x 9 columns]

data.plot(kind = "box",subplots = True, layout =
(6,2),figsize=(15,20))

fare_amount            AxesSubplot(0.125,0.772143;0.352273x0.107857)
pickup_longitude     AxesSubplot(0.547727,0.772143;0.352273x0.107857)
pickup_latitude        AxesSubplot(0.125,0.642714;0.352273x0.107857)
dropoff_longitude    AxesSubplot(0.547727,0.642714;0.352273x0.107857)
dropoff_latitude       AxesSubplot(0.125,0.513286;0.352273x0.107857)
passenger_count      AxesSubplot(0.547727,0.513286;0.352273x0.107857)
distance               AxesSubplot(0.125,0.383857;0.352273x0.107857)
dtype: object
```




```
# Using the InterQuartile Range to fill the values
def remove_outlier(df1 , col):
    Q1 = df1[col].quantile(0.25)
    Q3 = df1[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_whisker = Q1 - 1.5 * IQR
    upper_whisker = Q3 + 1.5 * IQR
    data[col] = np.clip(df1[col] , lower_whisker , upper_whisker)
    return df1

def treat_outliers_all(df1 , col_list):
    for c in col_list:
        df1 = remove_outlier(data , c)
    return df1

cols = ['fare_amount', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'passenger_count',
```

```
'distance']
data = treat_outliers_all(data , cols)

data.plot(kind = "box",subplots = True, layout =
(6,2),figsize=(15,20))
```

```
fare_amount            AxesSubplot(0.125,0.772143;0.352273x0.107857)
pickup_longitude     AxesSubplot(0.547727,0.772143;0.352273x0.107857)
pickup_latitude        AxesSubplot(0.125,0.642714;0.352273x0.107857)
dropoff_longitude    AxesSubplot(0.547727,0.642714;0.352273x0.107857)
dropoff_latitude       AxesSubplot(0.125,0.513286;0.352273x0.107857)
passenger_count      AxesSubplot(0.547727,0.513286;0.352273x0.107857)
distance               AxesSubplot(0.125,0.383857;0.352273x0.107857)
dtype: object
```




```
data.shape
```

```
(200000, 9)
```

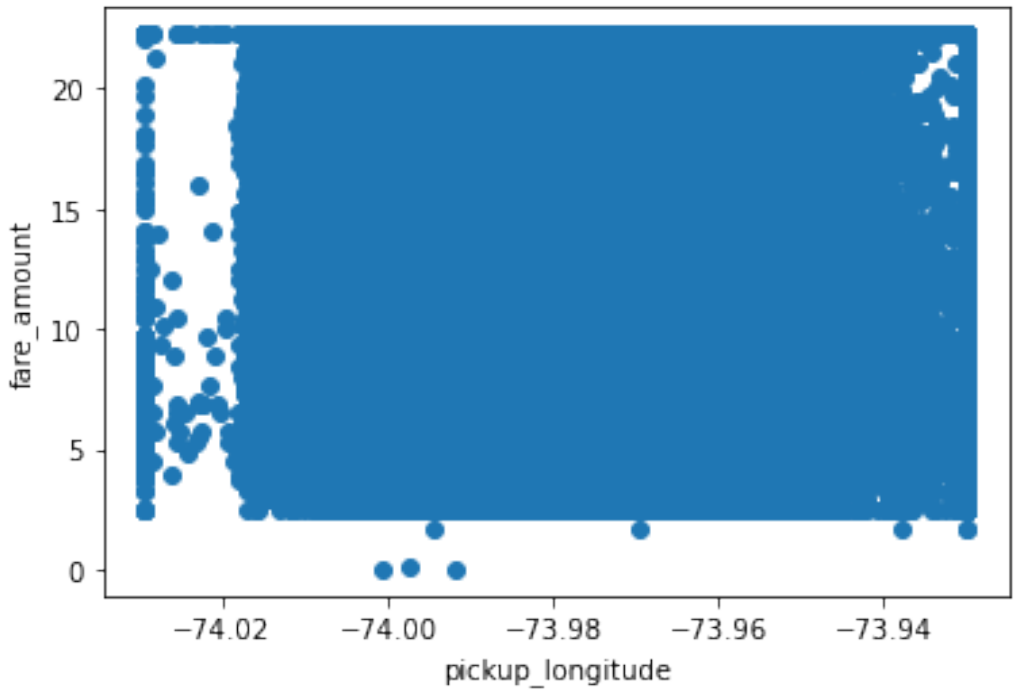
```
data.isnull().sum()
```

```
fare_amount          0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
month                0
hour                 0
distance             0
dtype: int64
```

VISUALIZATION

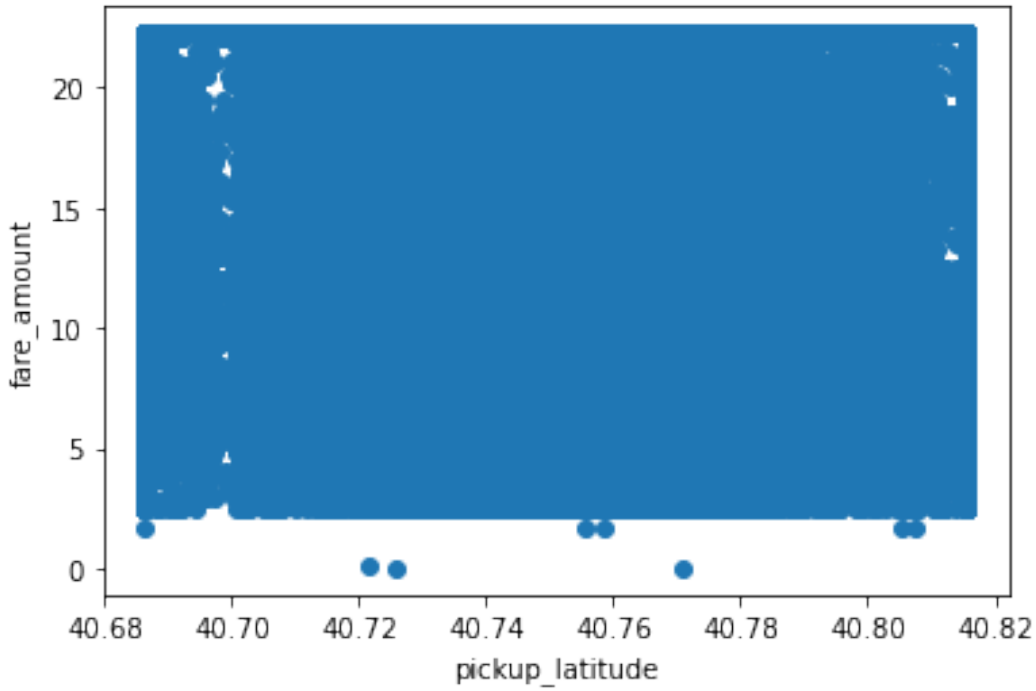
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.scatter(data['pickup_longitude'], data['fare_amount'])
plt.xlabel("pickup_longitude")
plt.ylabel("fare_amount")
```

```
Text(0, 0.5, 'fare_amount')
```

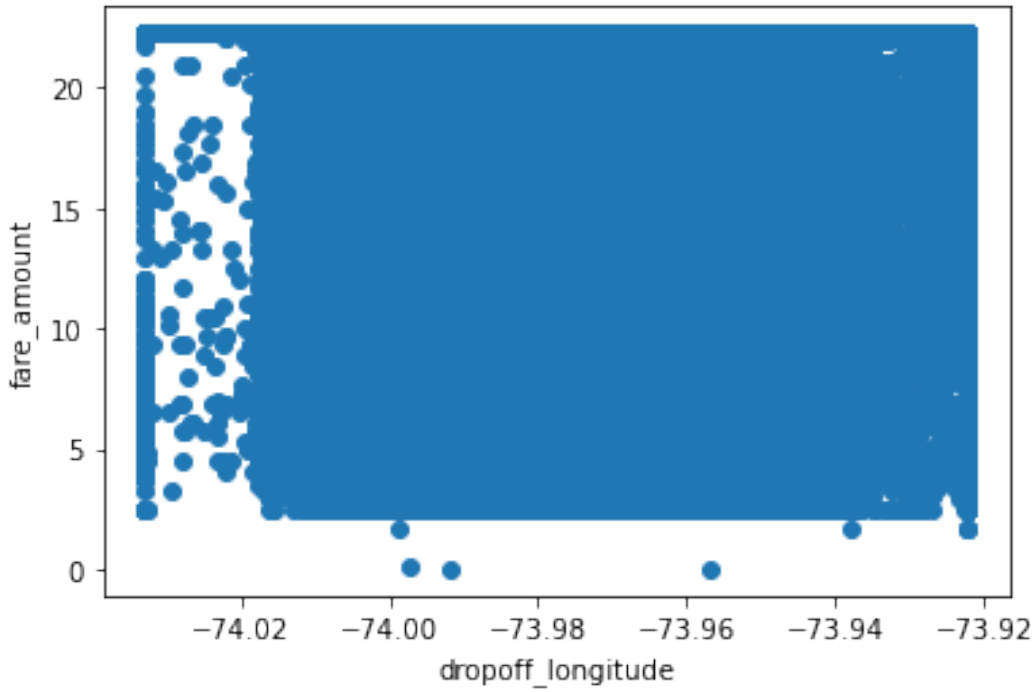



```
plt.scatter(data['pickup_latitude'], data['fare_amount'])
plt.xlabel("pickup_latitude")
plt.ylabel("fare_amount")
```

```
Text(0, 0.5, 'fare_amount')
```

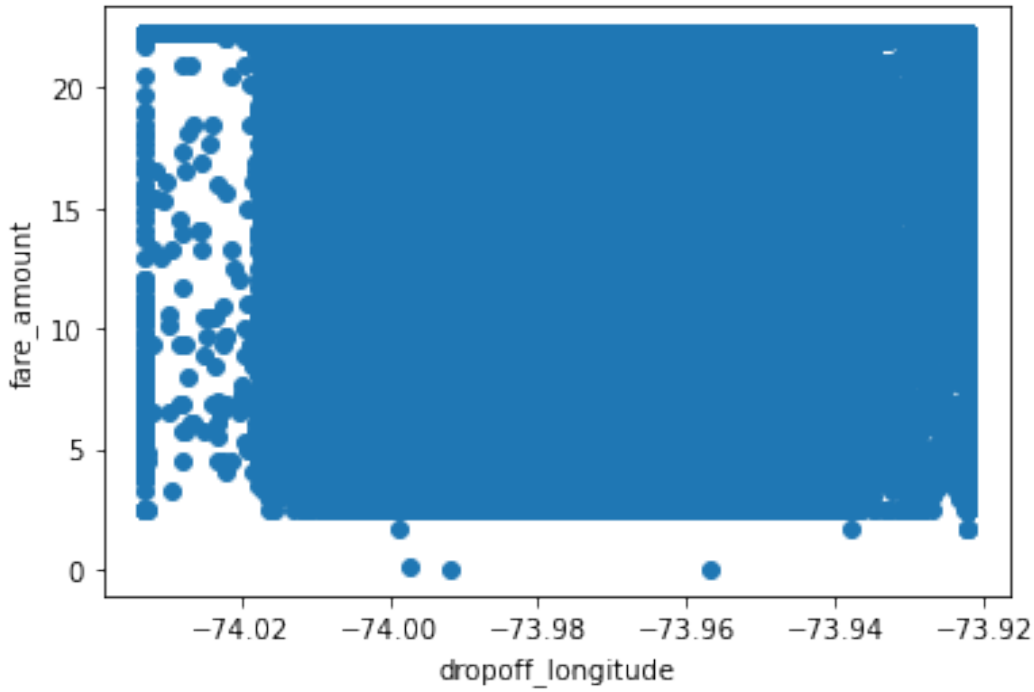



```
plt.scatter(data['dropoff_longitude'], data['fare_amount'])
plt.xlabel("dropoff_longitude")
plt.ylabel("fare_amount")

Text(0, 0.5, 'fare_amount')
```

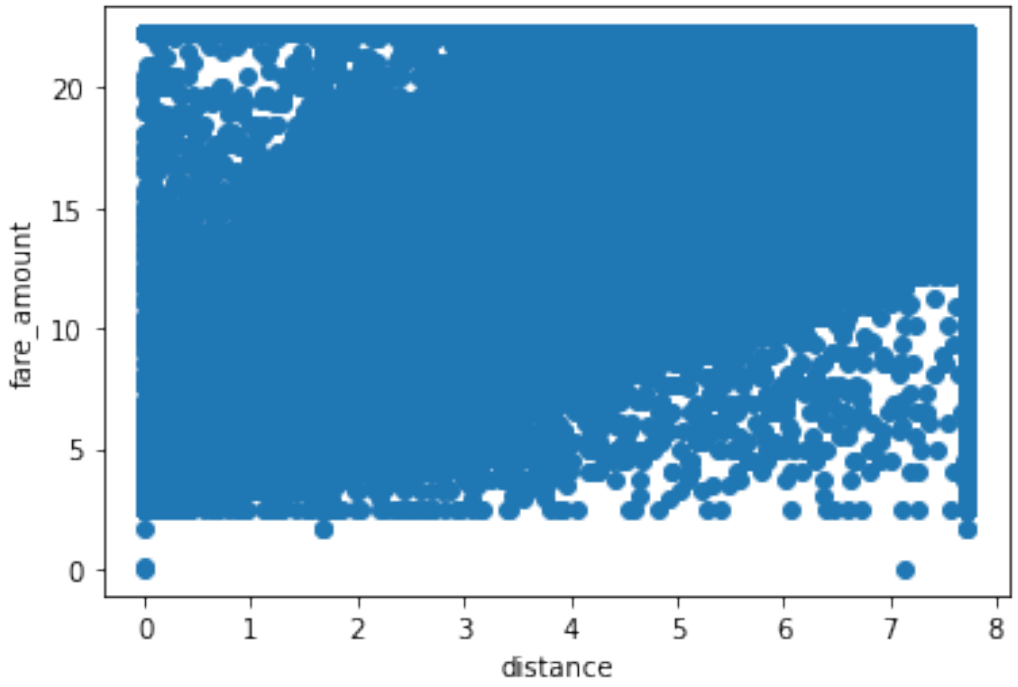



```
plt.scatter(data['dropoff_longitude'], data['fare_amount'])
plt.xlabel("dropoff_longitude")
plt.ylabel("fare_amount")
```

```
Text(0, 0.5, 'fare_amount')
```



```
plt.scatter(data['distance'], data['fare_amount'])
plt.xlabel("distance")
plt.ylabel("fare_amount")

Text(0, 0.5, 'fare_amount')
```



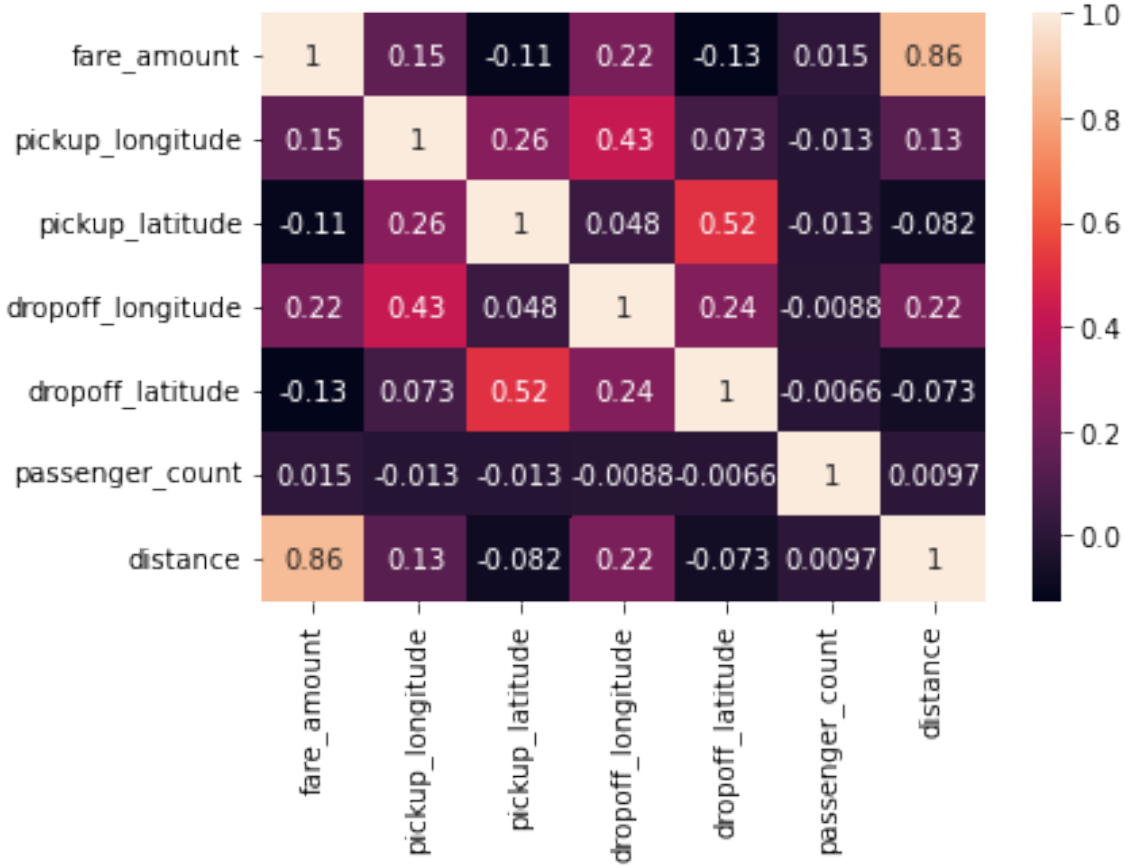

```
corr_matrix = data.corr()
corr_matrix

                   fare_amount  pickup_longitude  pickup_latitude  \
fare_amount           1.000000          0.154189        -0.110927
pickup_longitude      0.154189          1.000000         0.258704
pickup_latitude      -0.110927          0.258704         1.000000
dropoff_longitude     0.218764          0.425930         0.048264
dropoff_latitude     -0.125988          0.072689         0.515985
passenger_count       0.014649         -0.012819        -0.013306
distance              0.858832          0.134118        -0.082360

                   dropoff_longitude  dropoff_latitude
passenger_count  \
fare_amount                 0.218764         -0.125988
0.014649
pickup_longitude            0.425930          0.072689         -
0.012819
pickup_latitude             0.048264          0.515985         -
0.013306
dropoff_longitude           1.000000          0.244958         -
0.008774
```

```
dropoff_latitude             0.244958           1.000000          -
0.006609
passenger_count             -0.008774          -0.006609
1.000000
distance                     0.224214          -0.072548
0.009658
```

```
                    distance
fare_amount         0.858832
pickup_longitude    0.134118
pickup_latitude    -0.082360
dropoff_longitude   0.224214
dropoff_latitude   -0.072548
passenger_count     0.009658
distance            1.000000
```

```
sns.heatmap(corr_matrix, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23d90029cd0>
```



Spliting the dataset

```
X = data.iloc[:, 1:]
y = data.iloc[:, 0]
print(X.shape, y.shape)
```

```
(200000, 8) (200000,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=42)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(170000, 8) (170000,) (30000, 8) (30000,)
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
linear_regression = LinearRegression().fit(X_train, y_train)

linear_regression.score(X_test, y_test)
```

```
0.7316136867718988
```

```
y_pred = linear_regression.predict(X_test)

result = pd.DataFrame()
result['Actual'], result['Predicted'] = y_test, y_pred
result.sample(10)
```

```
        Actual  Predicted
63108     11.0   9.709085
82890      4.0   4.895952
102306    12.1   9.741012
92646     14.0  13.664903
47424     12.9   9.483992
172670     7.5   7.203042
133321    14.0  13.618706
177665     8.1   9.410597
58919     13.3  16.056232
64404      6.9   7.110601
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('R Squared (R²):', np.sqrt(metrics.r2_score(y_test, y_pred)))
```

```
Mean Absolute Error: 1.8763278733128748
Mean Squared Error: 7.9583356668996945
Root Mean Squared Error: 2.821052226900398
R Squared (R²): 0.8553441919905101
```

Random Forest

```
from sklearn.ensemble import RandomForestRegressor
random_forest = RandomForestRegressor(n_estimators = 10, random_state
= 42)

random_forest.fit(X_train, y_train)
```

```
RandomForestRegressor(n_estimators=10, random_state=42)
```

```
random_forest.score(X_test, y_test)
```

```
0.7525809054770123
```

```
y_pred = random_forest.predict(X_test)

result = pd.DataFrame()
result['Actual'], result['Predicted'] = y_test, y_pred
result.sample(10)
```

	Actual	Predicted
108389	7.70	9.50
16131	9.00	9.33
138295	22.25	19.76
32169	11.50	9.28
172860	5.00	4.77
20218	11.70	14.53
136756	7.70	10.04
148663	5.30	5.20
13249	2.90	7.17
78967	11.00	9.70

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print('R Squared (R²):', np.sqrt(metrics.r2_score(y_test, y_pred)))
```

```
Mean Absolute Error: 1.7530684681716462
Mean Squared Error: 7.336604392865712
Root Mean Squared Error: 2.7086166936031595
R Squared (R²): 0.8675142105331833
```